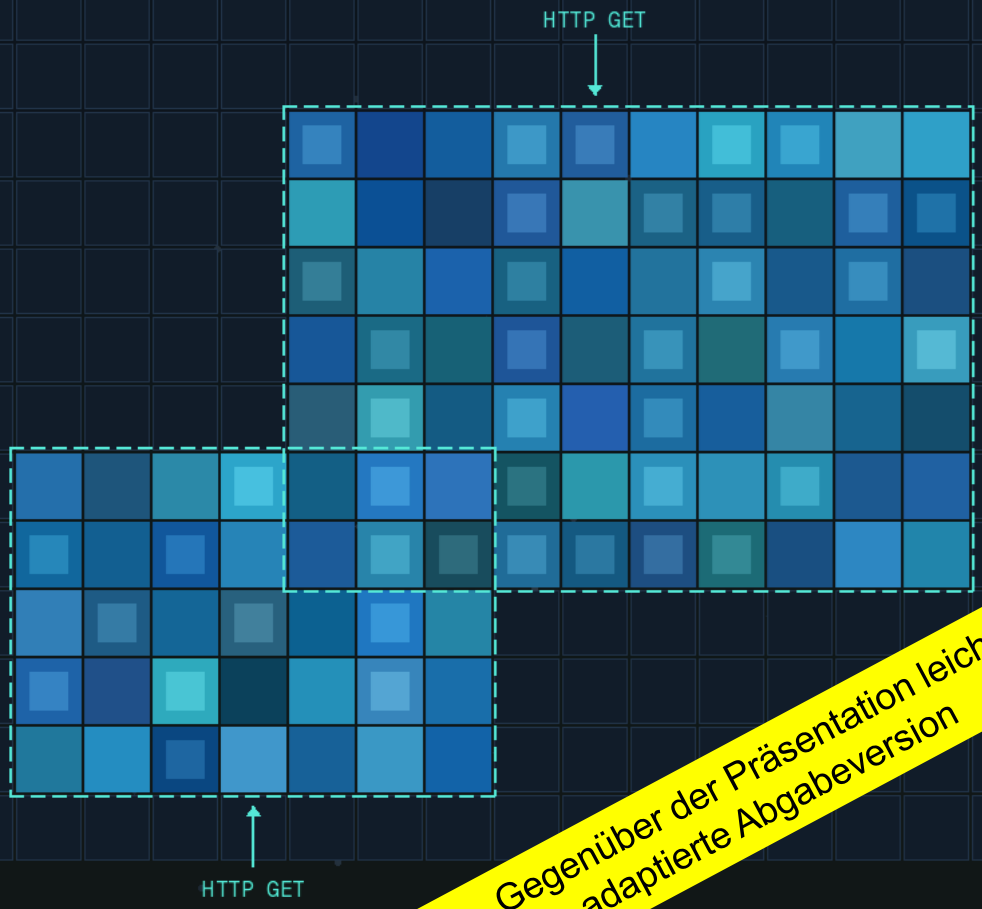
Cloud-Native Geospatial:

Was es kann, was nicht – und wann man es wirklich braucht

Ce que cette approche peut faire, ce qu’elle ne peut pas faire… et quand on en a vraiment besoin

Stefan Oderbolz, Ralph Straumann
KGK-CGC-Workshop
Olten, 03.09.2026



Gegenüber der Präsentation leicht adaptierte Abgabeversion

Agenda

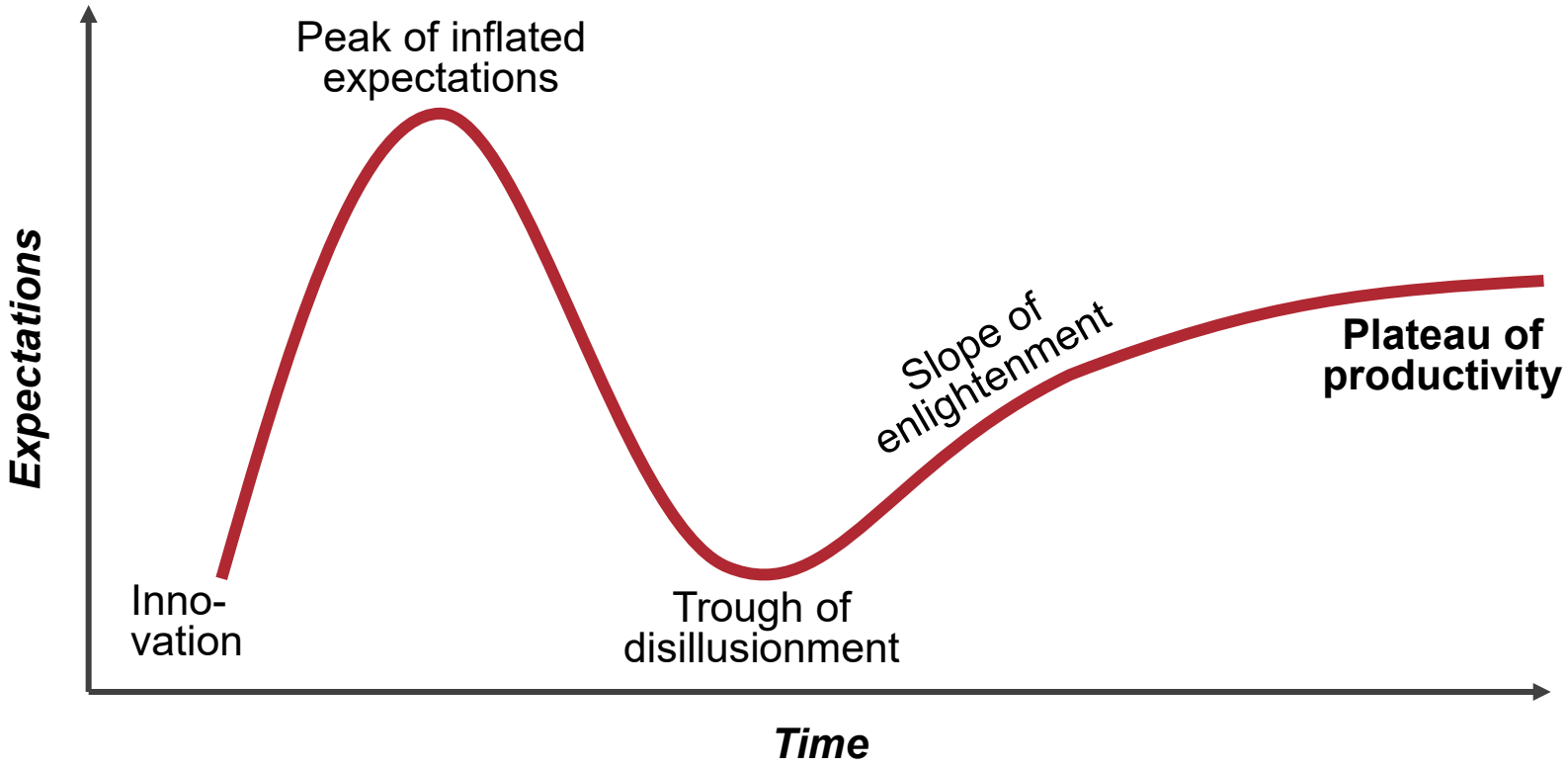
1. Einführung
2. CNG in der Praxis
3. Unsere Empfehlungen

1 CNG?

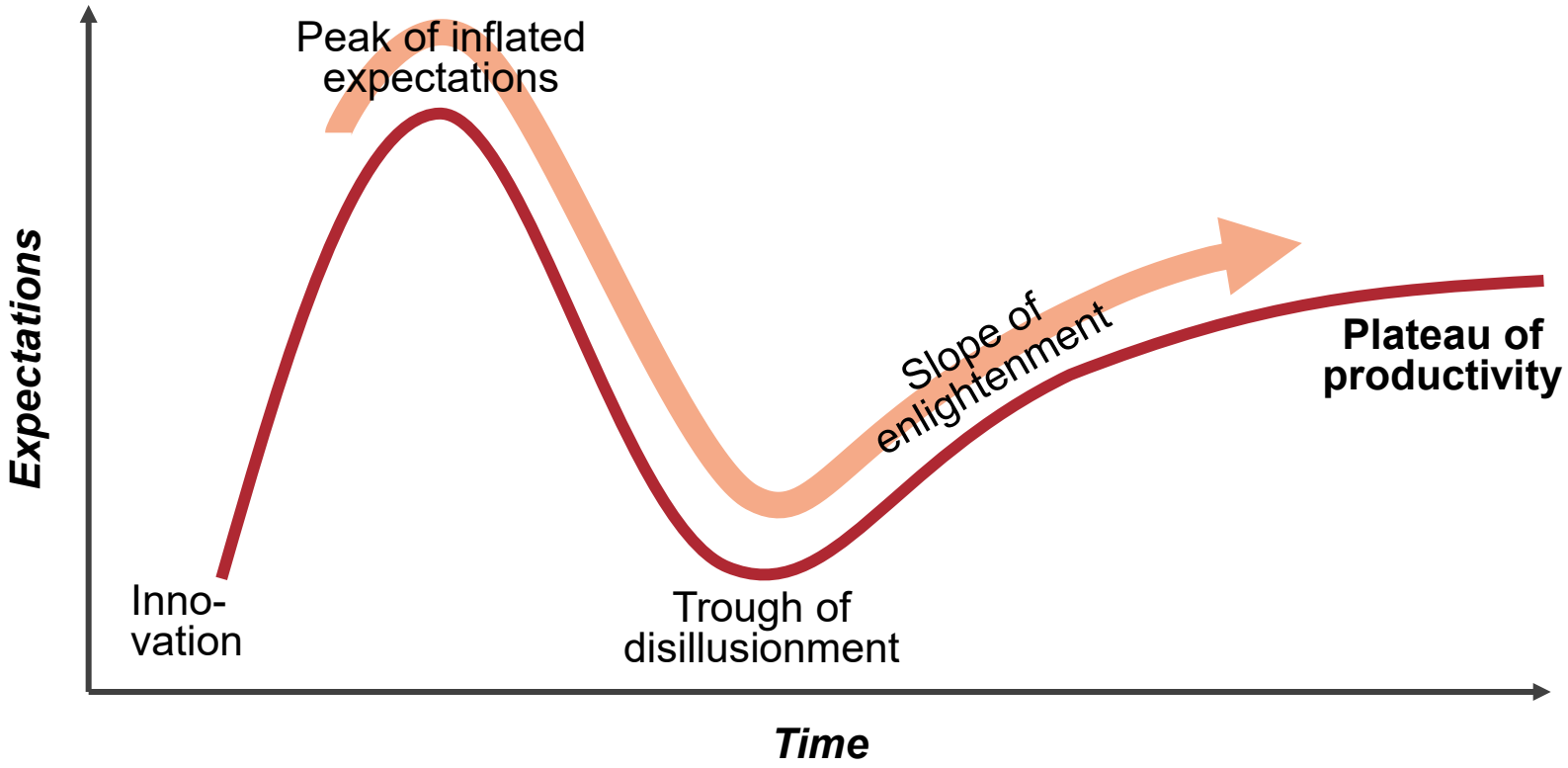
1 CNG?

«cloud-native geo»
«cloud-native geospatial»
#cloudnativegeo
#cng

Hype?


Peak of inflated expectations
Expectations
Slope of enlightenment
Plateau of productivity
Inno-vation
Trough of disillusionment
Time

Hype?


Peak of inflated expectations
Expectations
Slope of enlightenment
Plateau of productivity
Inno-vation
Trough of disillusionment
Time

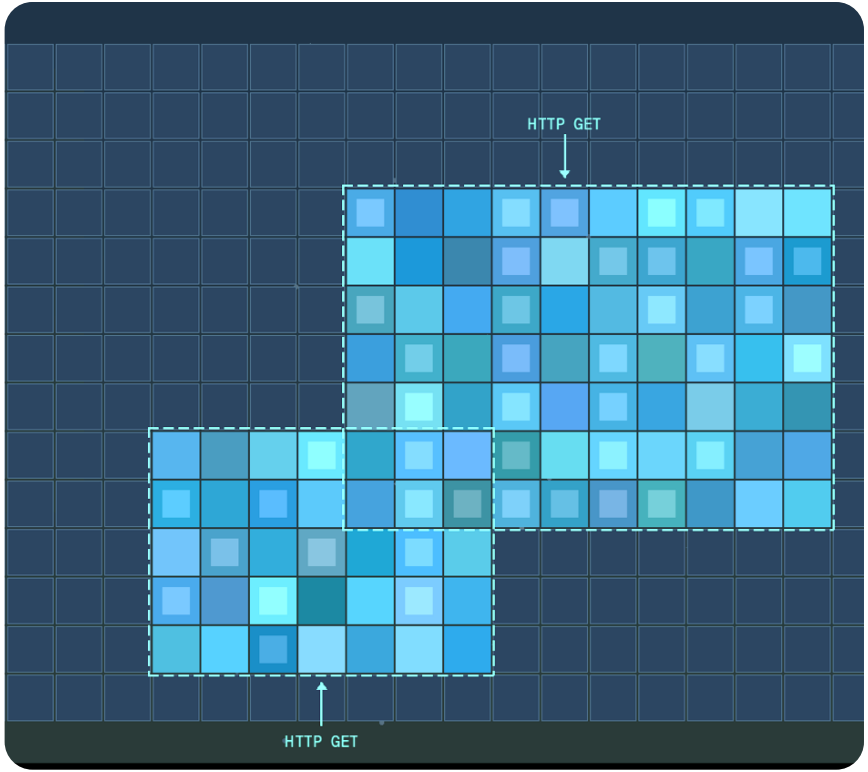
A **cloud-native dataset** is one with small addressable chunks via files, internal tiles, or both.

Formats are read-oriented to support:
– partial reads
– parallel reads

CNG Forum

CNG: Technologische Basis

- Daten aufgeteilt in **kleine, adressierbare Blöcke**: «chunks», «tiles», «row groups»
- Einfach zugreifbare **Metadaten in der Datei** selbst *(single read)*
- Partielle Abfragen via **HTTP Range-Requests**



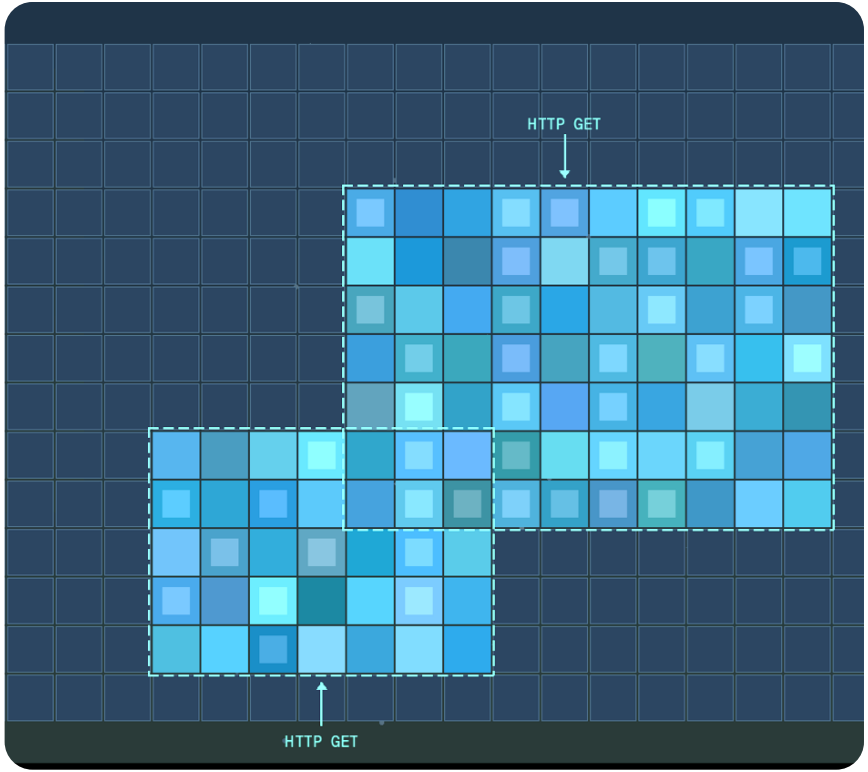
CNG: Technologische Basis

- Daten aufgeteilt in **kleine, adressierbare Blöcke**: «chunks», «tiles», «row groups»
- Einfach zugreifbare **Metadaten in der Datei** selbst *(single read)*
- Partielle Abfragen via **HTTP Range-Requests**

«An HTTP Range request asks the server to send **parts of a resource** back to a client.

Range requests are useful for various clients, including media players that support random access, **data tools that require only part of a large file** (…)»

MDN



Beispiel: **Cloud-Optimized GeoTIFF**

- Radiant Earth-Frontend
- Basis SWISSIMAGE
- 3-Kanäle: RGB
- Kanton Thurgau
- 10 cm Auflösung
- Via `gdalbuildvrt`, `gdalwarp` und `gdal_translate`
- *Eine* Datei: 14.84 GB

Beispiel: **Cloud-Optimized GeoTIFF**

- Radiant Earth-Frontend
- Basis SWISSIMAGE
- 3-Kanäle: RGB
- Kanton Thurgau
- 10 cm Auflösung
- Via `gdalbuildvrt`, `gdalwarp` und `gdal_translate`
- *Eine* Datei: 14.84 GB

```
[07/Oct/2024:11:40:22 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:22 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:22 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:22 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:22 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:22 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:23 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:24 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:24 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:24 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:24 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:24 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:24 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:24 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
[07/Oct/2024:11:40:24 +0200] "GET /[...]mosaic_10cm_cog.tif HTTP/2" 206
```

HTTP Response Status Code: 206 «Partial Content»

Übersicht über CNG-Formate

Format	Alias	Domäne	Version, Status 2026-09-03
Cloud-Optimized GeoTIFF	COG, COGTIFF, CoGeoTIFF	 Rasterdaten	1.0 als OGC-Standard
GeoParquet (und Parquet)	(GPQ)	 Vektordaten – spaltenorientiert, analytisch	2.0, OGC Standards Working Group (OGC SWG)
PMTiles		 Vektor- oder  Rasterdaten	3.0

Übersicht über CNG-Formate

Format	Alias	Domäne	Version, Status 2026-09-03
Cloud-Optimized Point Cloud	COPC	 Vektordaten: LiDAR-Punktwolken	1.0
FlatGeobuf	FGB	 Vektordaten – zeilenorientiert, streaming-optimiert	? – Kandidat für OGC Community Standard in 2023
Zarr (und GeoZarr)		 N-dimensionale Arrays	3.0, 2.0: OGC Community Stand.
Cloud-Optimized NetCDF4 / HDF5	(CONetCDF, COH5)	 N-dimensionale Arrays	beide OGC-Standards, aber nicht «cloud-optimized»


Die Technik ist geklärt.
Jetzt die Frage, die zählt.

Die Technik ist geklärt.
Jetzt die Frage, die zählt.

«Welches Problem wollen wir lösen?»

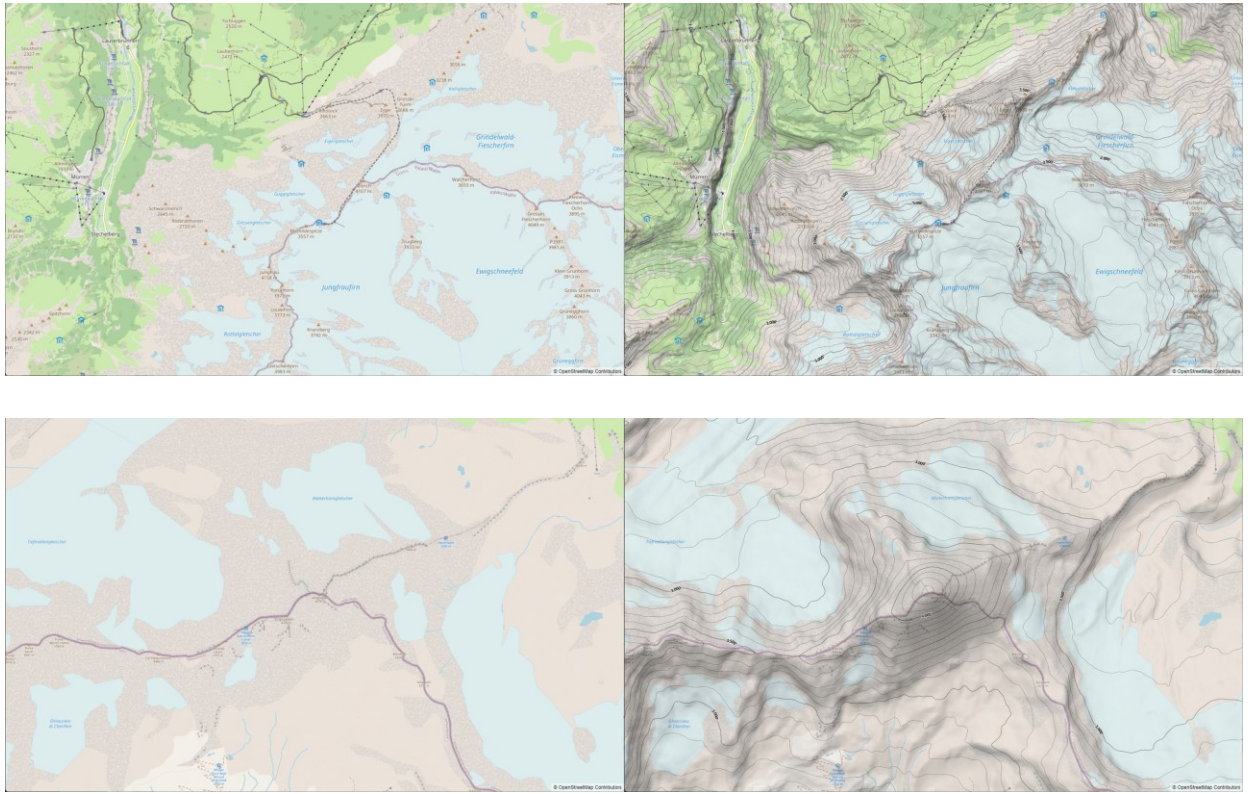
2 CNG in der Praxis

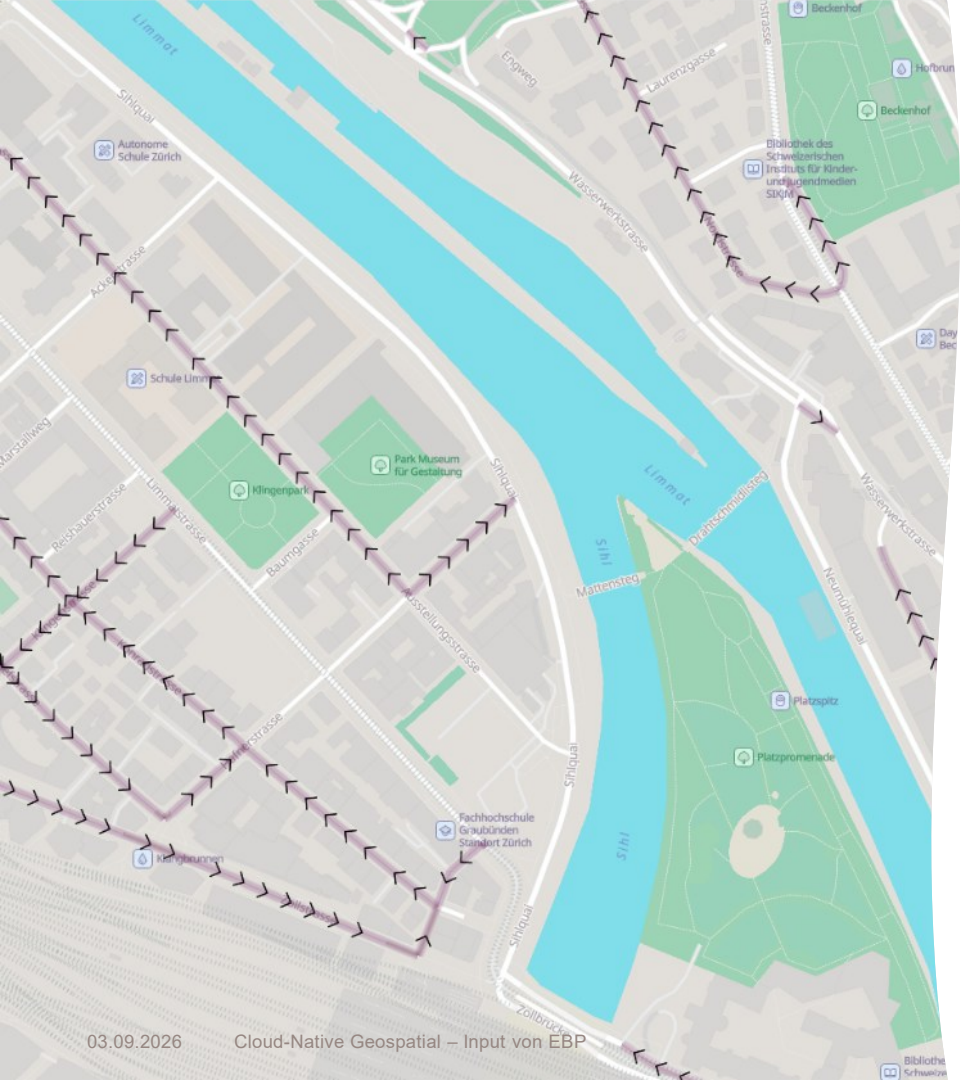
Praxisbeispiel 1: **DIY-Höhenlinien aus der Cloud**

Höhenlinien und Relief für eine Webanwendung

- Copernicus-DEM
- GDAL merge
- Rasterio rgbify (Höhe RGB-codiert)
- MBTiles (Zoom 10–13)
- pmtiles CLI

Resultat: Eine PMTiles-Datei, 3 GB, auf AWS S3, MapLibre GL JS rechnet Höhenlinien, Hillshade und 3D-Terrain *on-the-fly* im Client





Praxisbeispiel 2: **Alle Einbahnstrassen Europas**

Einbahnstrassen ganz Europas für ein Forschungsprojekt vergleichbar darstellen.

- OpenStreetMap via Overpass API (länderweise)
- GeoJSON
- Tippecanoe
- Tile-join

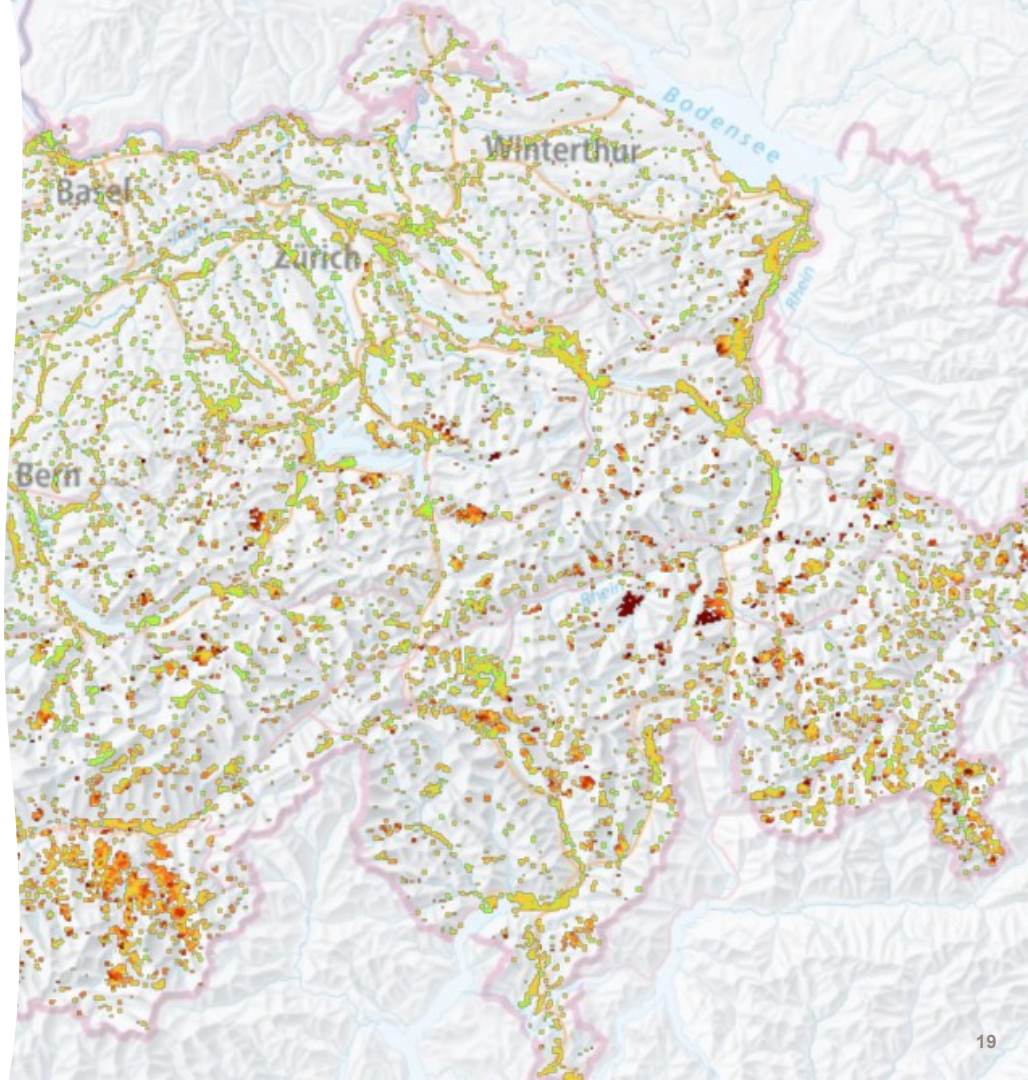
Resultat: Eine PMTiles-Datei (4.5 GB), MinIO, MapLibre GL mit Protomaps-Basiskarte (40 GB)

Praxisbeispiel 3: **InSAR-Daten auf der Naturgefahrenplattform GIN**

GIN integriert erstmals Satellitendaten (interferometrisches SAR) zur Früherkennung von Bodenbewegungen und Rutschungen.

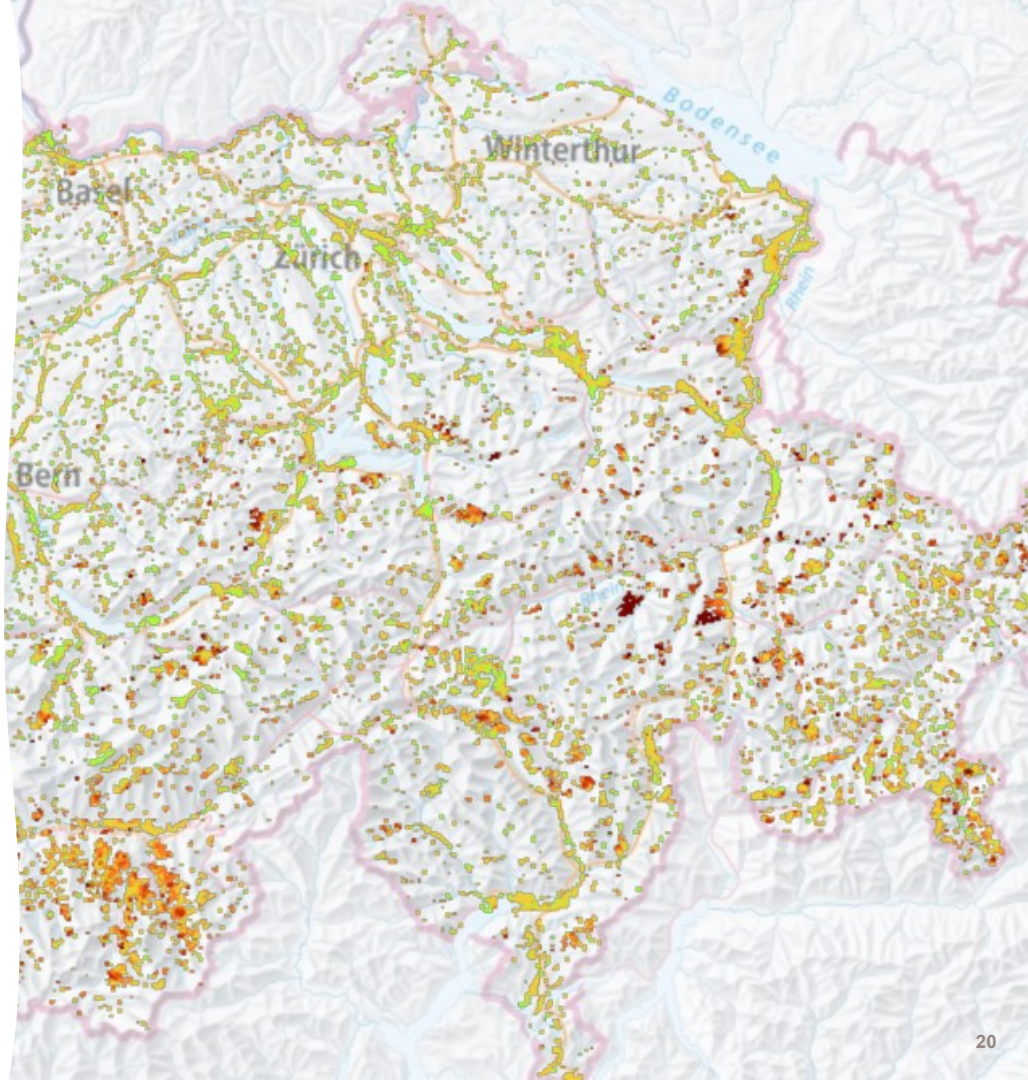
7 Produkte, 30 MB – 25 GB pro Datensatz, Millionen *Messpunkte* mit Zeitreihen aus dutzenden Werten.

→ Millionen von Geometrien mit Attributdaten. Und das Datenvolumen wächst jährlich.



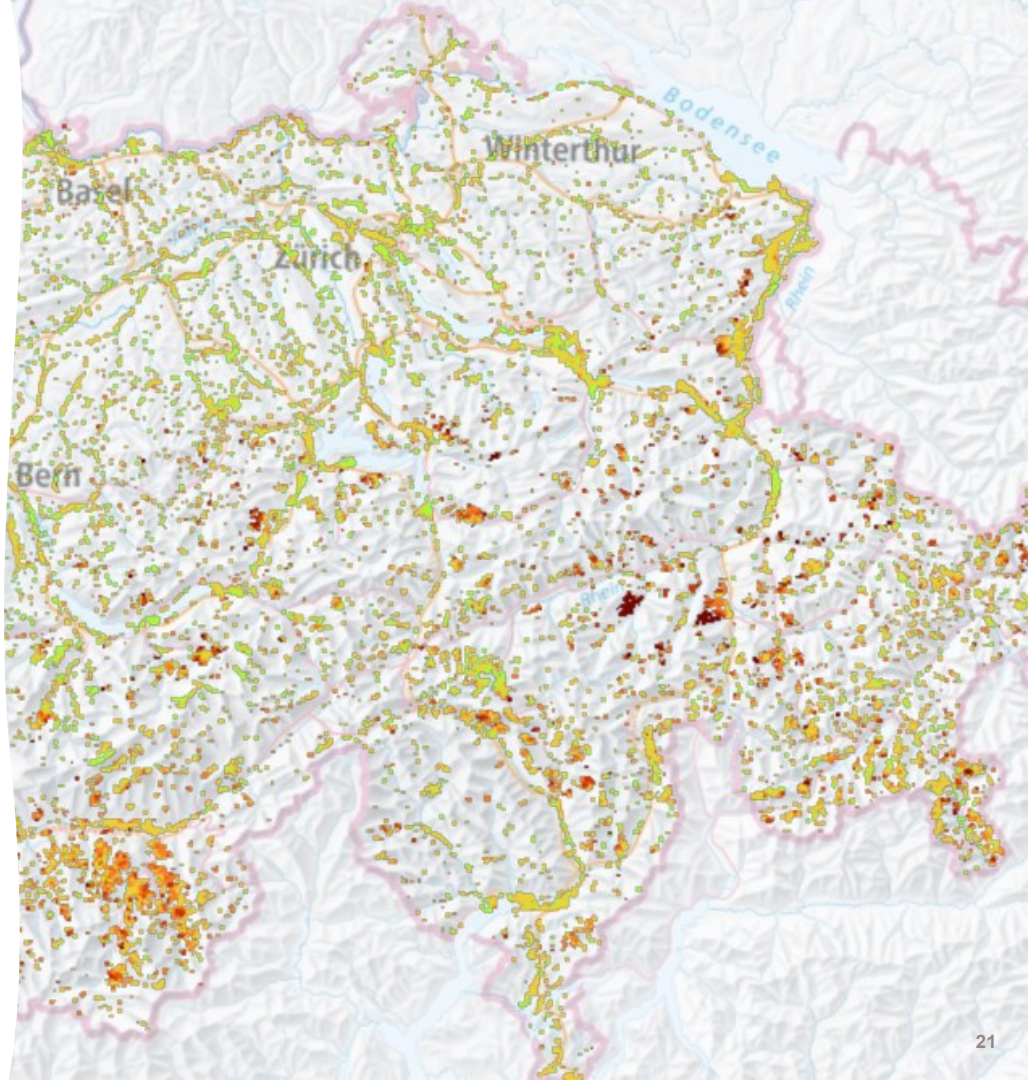
Praxisbeispiel 3: **InSAR-Daten auf der Naturgefahrenplattform GIN**

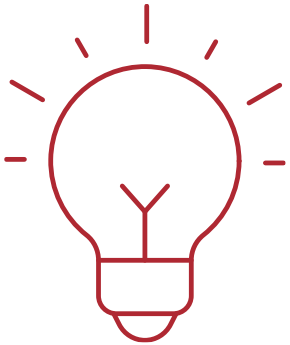
- **Update** ist trivial: Dateien werden jährlich ausgetauscht (keine inkrementellen Updates)
- **Visualisierung:** COG für die Übersicht, PMTiles für hohe Zoomstufen
- **Abfragen**: DuckDB liest Parquet-Dateien. Latenz war initial zu hoch, daher Wechsel von S3 auf EFS.



Praxisbeispiel 3: **InSAR-Daten auf der Naturgefahrenplattform GIN**

- **Knowhow**: Vektor→Raster-Konversion für die Ableitung von COGs verlangte viele **fachliche Entscheide**:
  - Auflösung
  - Farbskala
  - gute Aggregationsregeln, damit Risiken nicht «verschwinden»
- **Support**: PMTiles in OpenLayers hat immer noch einige «**Kinderkrankheiten**» (z.B. Artefakte an Kachelgrenzen)



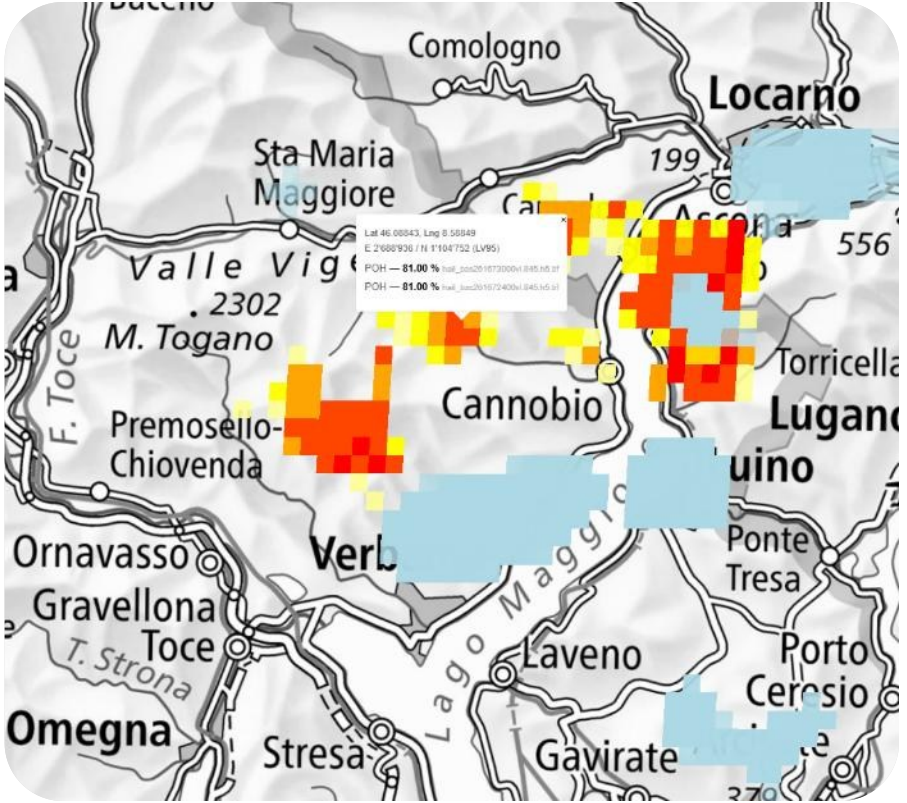


Die **Datenstruktur** entscheidet stärker über die **Performance** als das Format.

Praxisbeispiel 4: **Wetterdaten für eine Versicherung**

Im Unternehmenskontext steht cloud-native nicht allein, sondern neben einer etablierten GIS-Landschaft.

- MeteoSchweiz-Radardaten für Hagel (POH, MESHS) und Niederschlags-verteilung (CombiPercip) stehen via als NetCDF oder HDF5 zur Verfügung.
- Bereitstellung der Daten als COG für interne Analysen, **Integration in be-stehende ArcGIS Mosaic-Datasets.**
- Zugang via internen **STAC**
- Die Daten der **letzten ~14 Tage sind im Hot-Tier** verfügbar, dann im **Cold-Tier**, die letzten ~30 Jahre im **Archive-Tier**.



3 Unsere Empfehlungen

Wann cloud-native Sinn ergibt…

Datengrösse

Zu gross zum praktischen Herunterladen

InSAR, 25 GB pro Produkt

Publikation

Breite Verfügbarkeit ohne eigene Server-Infrastruktur

Einbahnstrassen Europas

Stabilität

Die Daten ändern selten oder nie

Luftbilder, InSAR-Jahresstände

Ökosystem

Daten sollen in Data Pipelines, Web-Apps und / oder Analytics

GeoParquet mit DuckDB, Modern Data Stack (MDS)

… und wann nicht

«in der Cloud» ≠ «cloud-native»

Nein-Fälle:

- Der Datensatz passt auf eine lokale Festplatte.

  → GeoPackage, ev. normales GeoTIFF, PostGIS

- Die Analyse läuft auf einem Rechner oder einem Server.
- Die Daten ändern laufend und fein-granular.
- Zugriffskontrolle ist zentral.

… und wann nicht

«in der Cloud» ≠ «cloud-native»

Nein-Fälle:

- Der Datensatz passt auf eine lokale Festplatte.

  → GeoPackage, ev. normales GeoTIFF, PostGIS

- Die Analyse läuft auf einem Rechner oder einem Server.
- Die Daten ändern laufend und fein-granular.
- Zugriffskontrolle ist zentral.

… oder: was man sich einhandelt

Stolperfallen aus der Praxis

- **Latenz-Akkumulation** – Jeder Request kostet Latenz. Viele sequenzielle Requests über räumliche Distanzen sind langsamer als eine lokale DB.
- **Formatwahl entscheidet** – FlatGeobuf hat Spatial Index, ist aber schwach bei analytischen Queries, GeoParquet ist stark für Analytik, hat aber (noch) keinen eingebauten Spatial Index.
- **Ungleiche Tool-Reife** – Python (`rasterio`, `fsspec`, `xarray`, `dask`) ist ausgereift; R, JavaScript und Desktop-GIS haben Lücken.
- **Neue Optimierungskomplexität** – Ein simples `SELECT *` kann in DuckDB plötzlich ineffizient und teuer sein.
- **Der Betrieb verschwindet nicht** – Object Storage will verwaltet, Berechtigungen konfiguriert, Konvertierungs-Pipelines gebaut und gepflegt werden.

Nicht:
Wie machen wir unsere Daten cloud-native?

Sondern:
Welches Problem wollen wir lösen?

Vielen Dank! – Fragen?

Stefan Oderbolz
+41 44 395 10 30
stefan.oderbolz@ebp.ch

Ralph Straumann
+41 44 395 11 35
ralph.straumann@ebp.ch

www.ebp.ch

digital.ebp.ch/cng